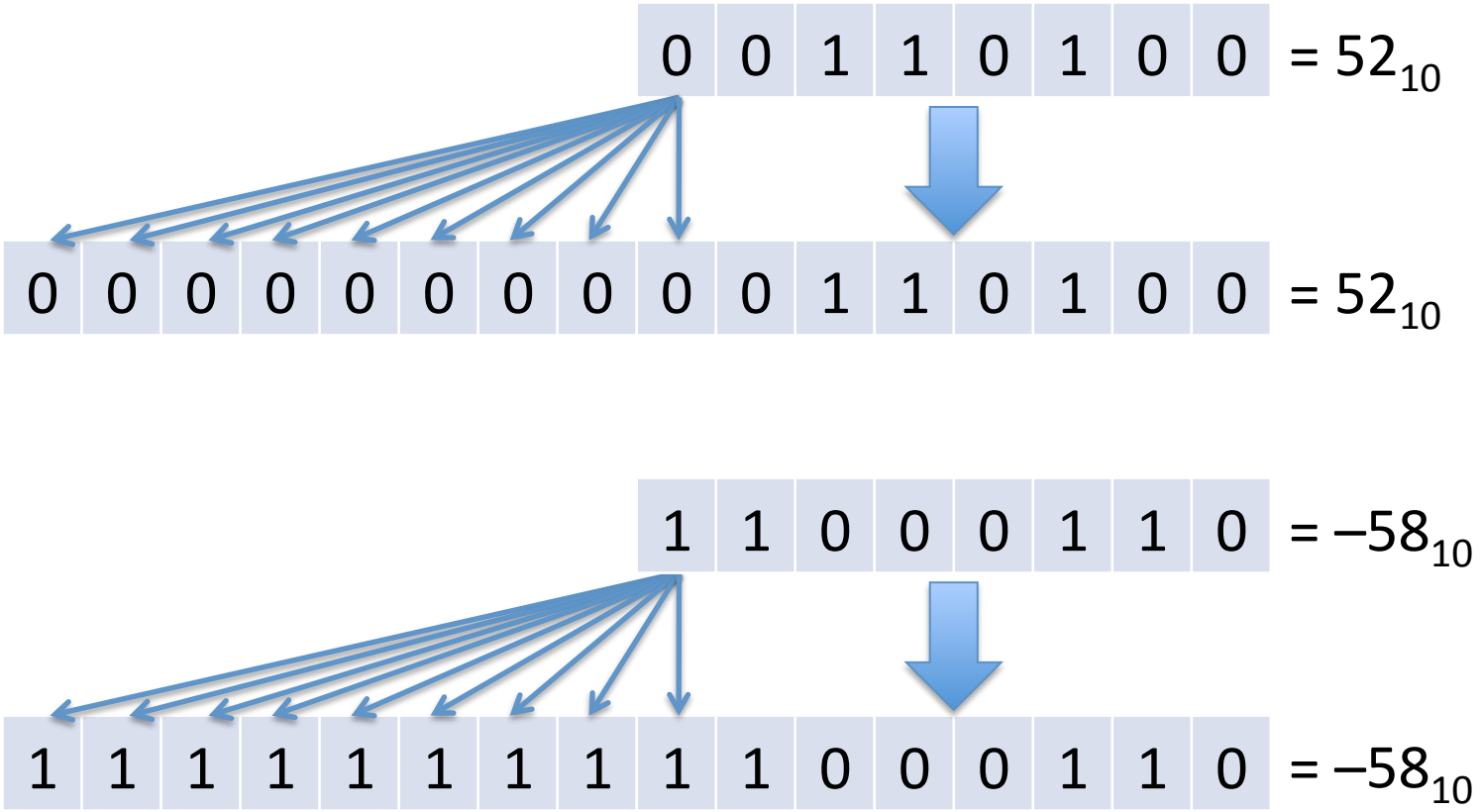
Assembly en Assemblers

Processoren

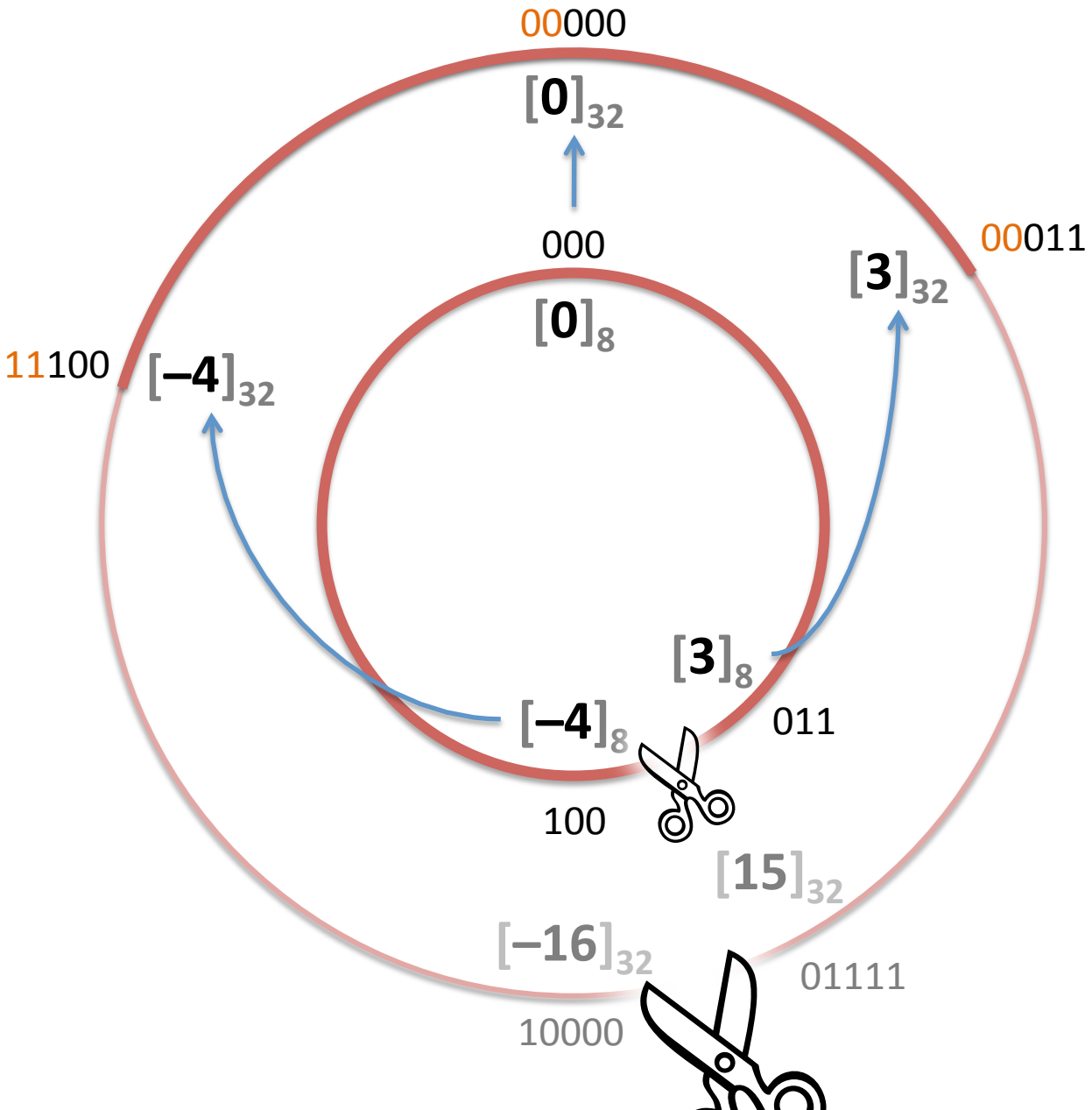
20 maart 2012

Sign extension

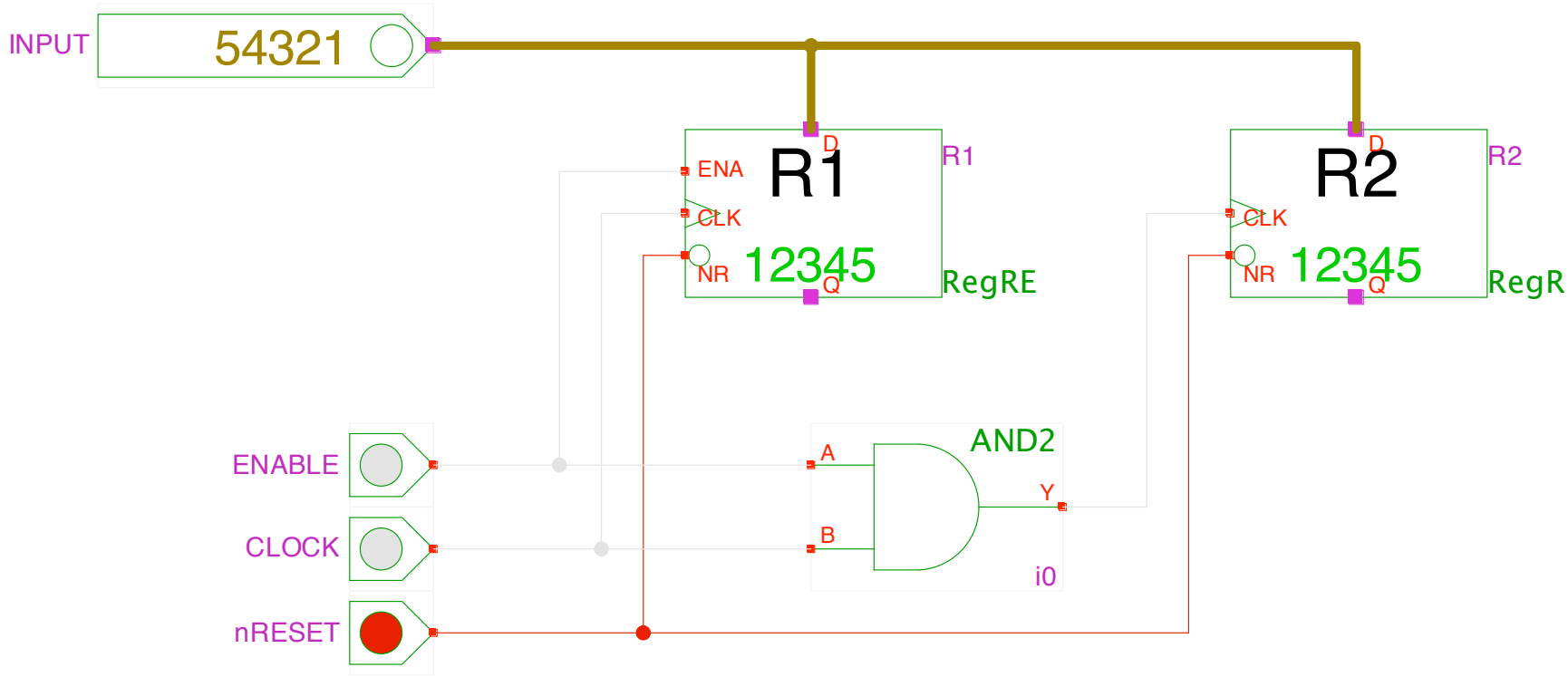
Hoe schaal je een getal in twee-complement op?



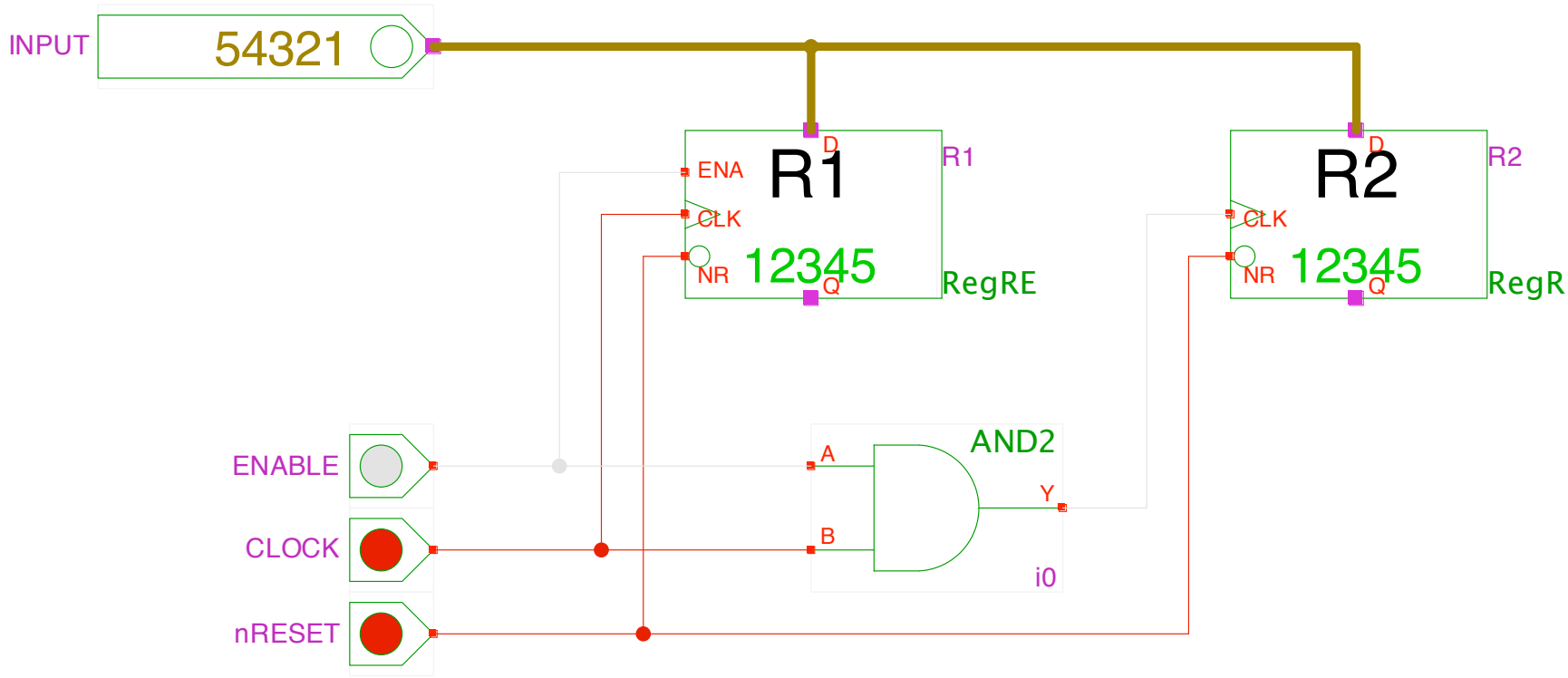
Sign extension



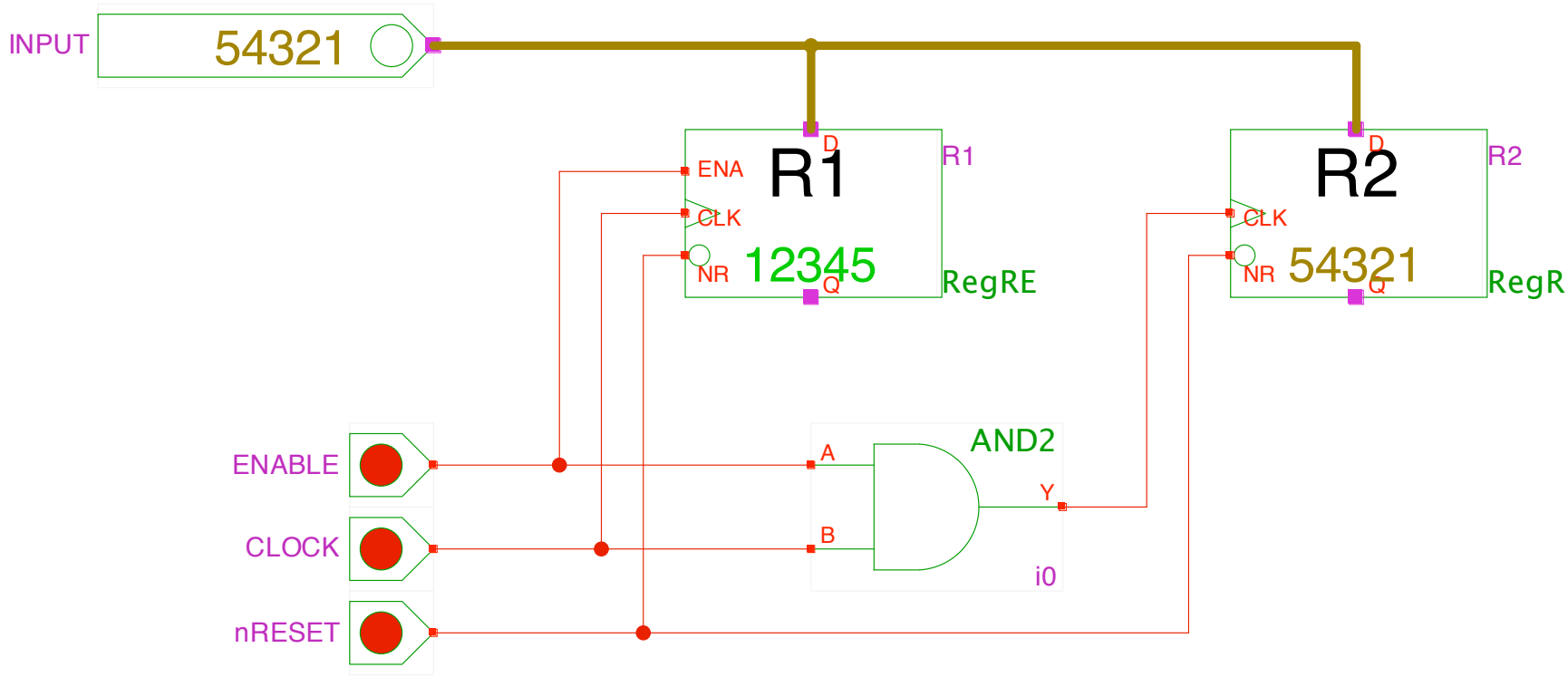
Enable- en klok-invoer



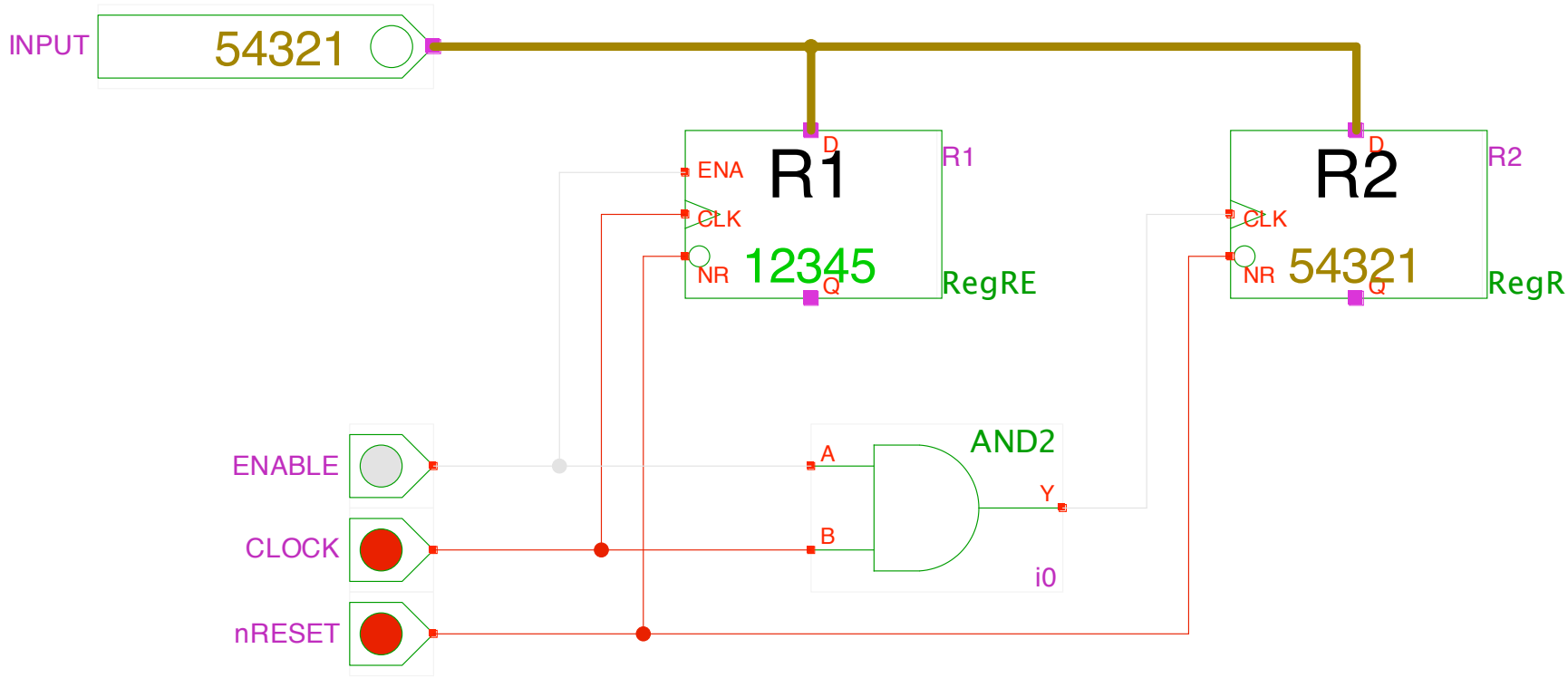
Enable- en klok-invoer



Enable- en klok-invoer



Enable- en klok-invoer



Enable- en klok-invoer

- De twee registers lijken equivalent,
  maar zijn het niet.
  - als eerst CLOCK van 0 naar 1 springt
    en daarna ENABLE, krijg je een verschil.
  - ENABLE=1 gedurende een paar nanoseconden
    is voldoende.
- De klok als enable misbruiken is onveilig.
- Gebruik registers/flipflops mét enable-invoer!

Positieve en negatieve signalen

- “positive logic”: signaal = 1 → actief
- “negative logic”: signaal = 0 → actief

- voorbeelden van positive logic:
  ENABLE, FETCH
- voorbeelden van negative logic:
  nRE, nWE

Assembly en Assemblers

Processoren

20 maart 2012

Doel van vandaag

- Ik heb al losse eindjes over assembly verteld en een voorbeeldprogramma doorlopen.

- vandaag: algemeen + systematisch overzicht

Programmeertalen

- Machinetaal: bitpatronen in het geheugen
  - moeilijk te lezen
  - makkelijk fouten te maken
- Assembly: mnemonische instructienamen

  - kan één voor één in machinetaal vertaald worden
- Hogere programmeertaal:
  abstracte, wiskundige notatie

Voor- en nadelen van Assembly

t.o.v. machinetaal	t.o.v. hogere talen
+ makkelijker te onthouden + minder foutgevoelig	+ volledige controle van de CPU + efficiënte, kleine programma's − slechts voor één processortype − kleine instructies → foutgevoelig

Formaat van assembly-instructies

- per regel één instructie of pseudoinstructie
  - (label)
  - (pseudo)instructienaam
  - operanden
  - (commentaar)

Labels

- namen voor adressen in code of data
  - code: waarheen de processor kan springen

```
            J.O         error
            ...
            ...
error:      ...
```

  - data: globale variabelen

```
            READ    [length], R1
            ...
            ...
length:     .data   177
```

Instructies

- Naam: geeft aan wat de instructie doet
  - ADD optellen
  - READ (uit RAM) lezen
  - DAA (x86) decimal adjust after addition
  - kleine verschillen tussen namen (MOV/MOVE) volgens smaak van de ontwerper
- Instructie bepaalt aantal + formaat operanden

Pseudoinstructies

- regieaanwijzingen aan de assembler
  - waar moet het programma opgeslagen worden?
    ```
    .org        1024
    ```
  - datadefinitie, b.v.
    ```
    hellostring: .data      "Hello, world!", 0
    ```
  - macrodefinities
- lijken vaak erg op instructies

Voorbeeld: x86-assembly-programma

Label	Opcode	Operands	Comments
FORMULA:	MOV	EAX,I	; register EAX = I
	ADD	EAX,J	; register EAX = I + J
	MOV	N,EAX	; N = I + J
I	DD	3	; reserve 4 bytes initialized to 3
J	DD	4	; reserve 4 bytes initialized to 4
N	DD	0	; reserve 4 bytes initialized to 0

Wat doet een assembler?

- mnemonische namen vertalen
- adressen berekenen
- (vaak) macros

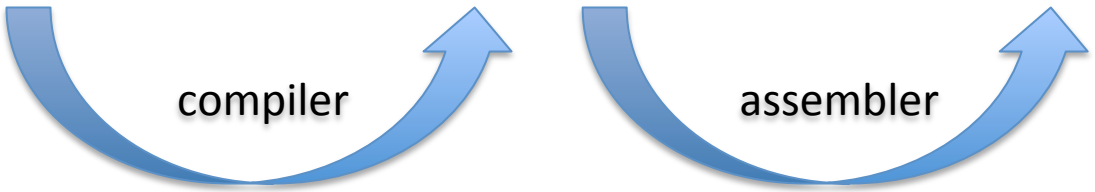
vaste afkortingen van een paar instructies

- pseudoinstructies

constanten opslaan, macrodefinities

Vertaalstappen

hogere taal	assembly		machinetaal
i := i + 3;	LOADHI	13, R2	800D
	ADD	7, R2, R2	06A7
	READ	[R2], R1	1100
	ADD	3, R1, R1	17A3
	WRITE	R1, [R2]	1900



- Soms zit de assembler in de compiler ingebouwd.

Twee-pass-assembler

- Probleem: forward reference
  = label-adres is soms nog niet bekend als het gebruikt wordt

- Oplossing: programma 2× lezen
  - 1e pass: symbooltabel berekenen

  - 2e pass: instructies vertalen
    en objectbestand aanmaken


Samen houden we het studentenleven draaiende, word bestuurslid!
19 T/M 23 MAART: WEEK VAN HET STUDENTBESTUUR
Woensdag 21 maart: Open Kamerdag
Donderdag 22 maart: Besturen iets voor jou?!
Voorlichtingsbijeenkomst
CC5 > 12.30 - 13.30 uur
www.ru.nl/studentbestuur
STICHTING NIJMEEGS
UNIVERSITEITSFONDS
Hulpfonds voor de Nijmeegse student
USR
Universitaire Studenten Raad
Radboud Universiteit Nijmegen


Bijbel
Alpha
Trainen
Studentenhuis
Cursus
StudentAlpha
Geloof
Discussie
Praatje
Hemel
God
Huiskamer
Jezus
Student
Kerk
Gesprek
College
Heilige Geest
Tentamen
Kroeg
Bidden
Borrel
Relaxed
Maaltijd
STUDENTALPHA.NL
studentalpha

Wat rest er na assembleren?

- Grote programma's opgedeeld in modules
  - elke module apart assembleren
  - na het assembleren
    voegt een **linker** de objectbestanden samen
    tot een programmabestand

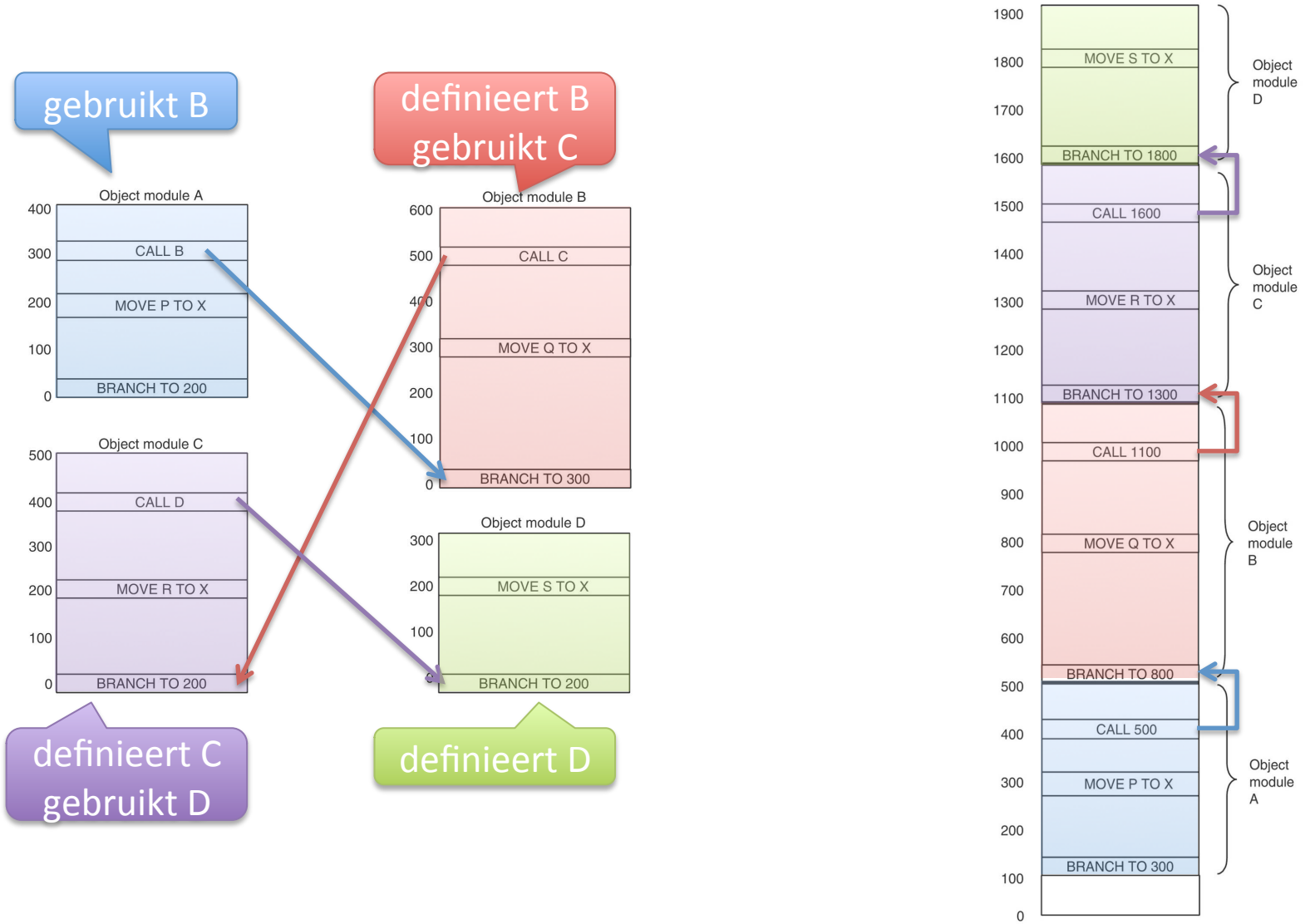
  - voordeel: bij kleine wijziging sneller

Taken van de linker

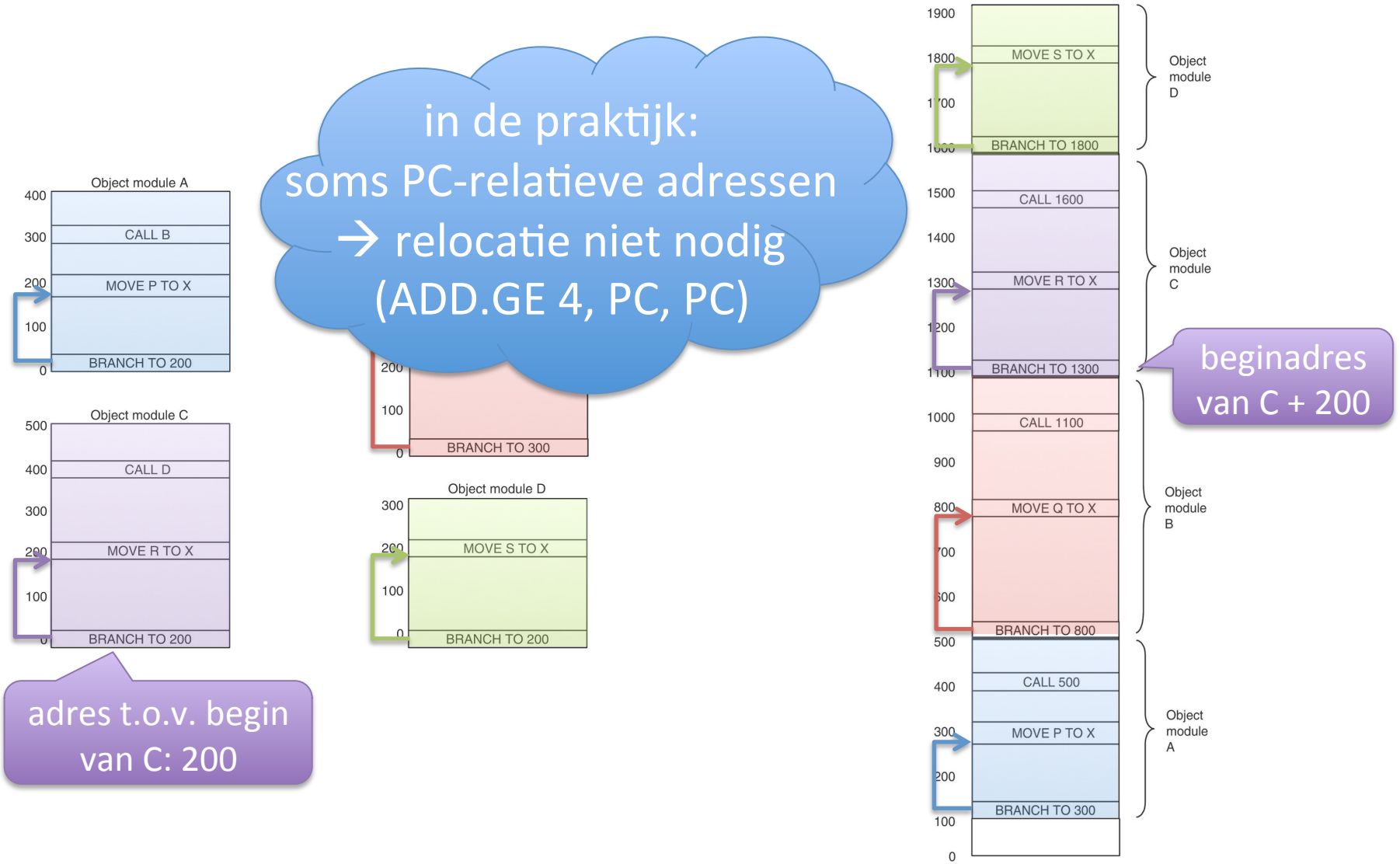
1. symbooltabellen vergelijken en samenvoegen
   - pseudoinstructies geven aan welke symbolen gedefinieerd of gebruikt worden

2. reloceren (eerste deel)
   - startadres van de modules kiezen
   - adressen daaraan aanpassen

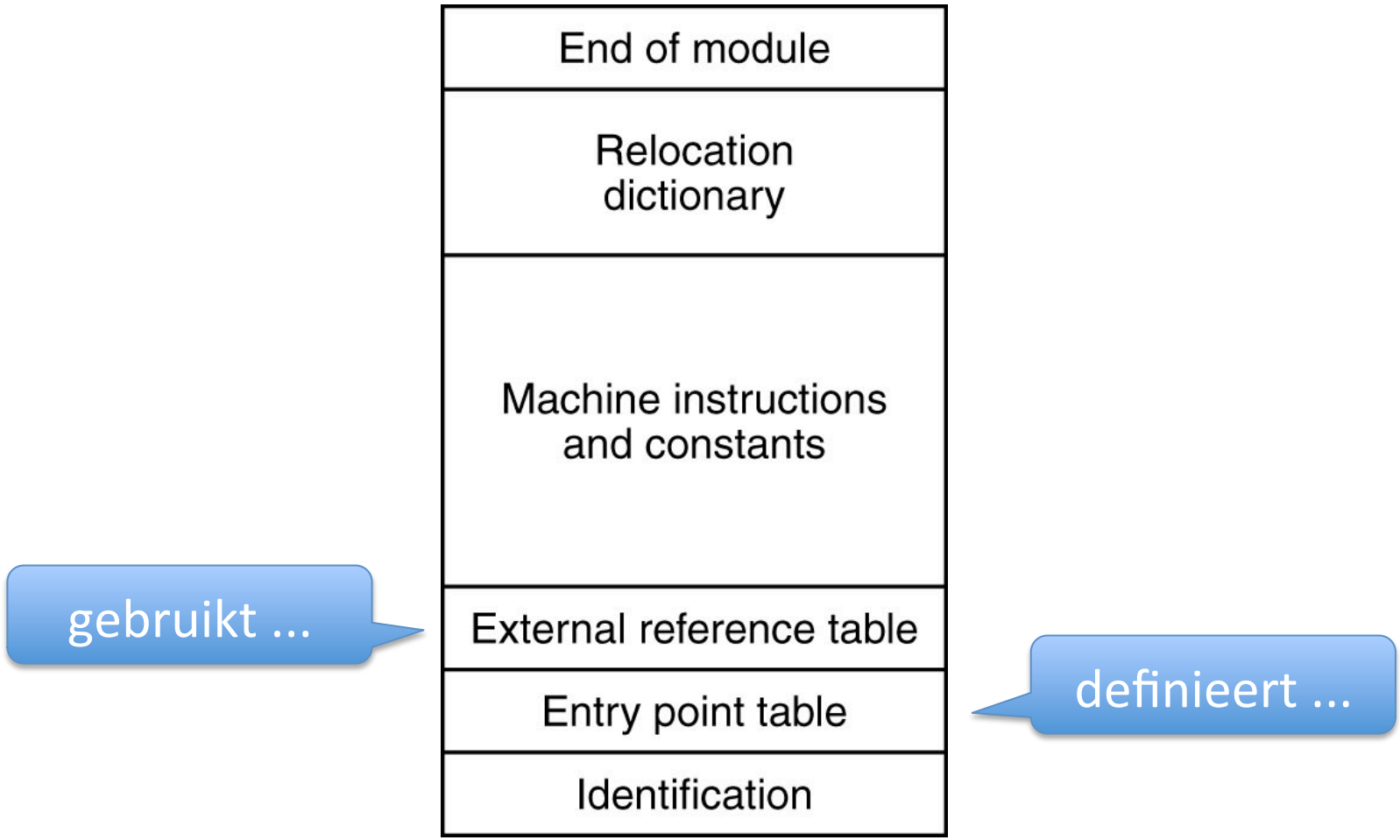
Symbooltabellen samenvoegen



Reloceren



Onderdelen van een objectbestand



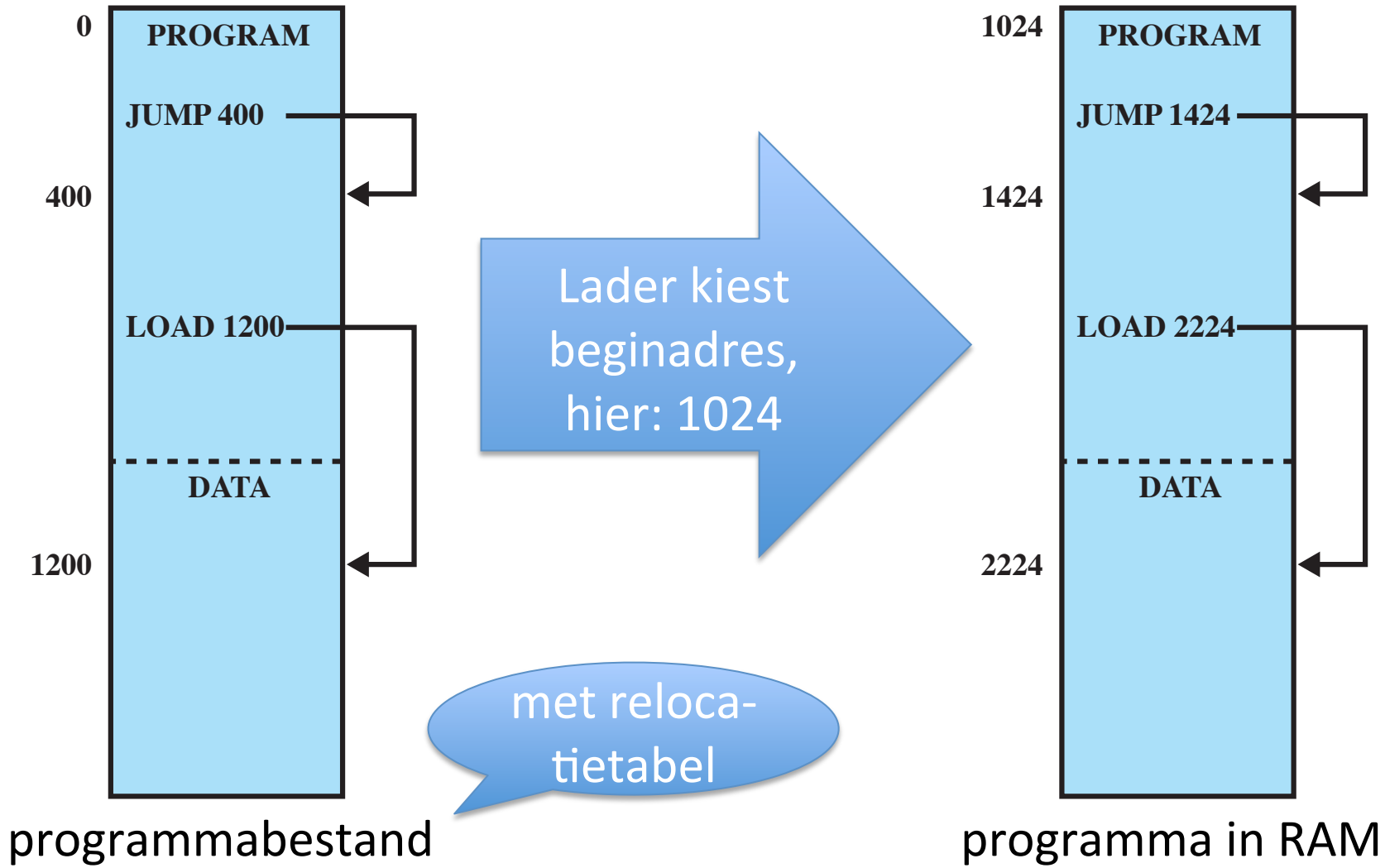
Programmabestand is nog geen programma

- **Lader** kopieert programmabestand naar RAM

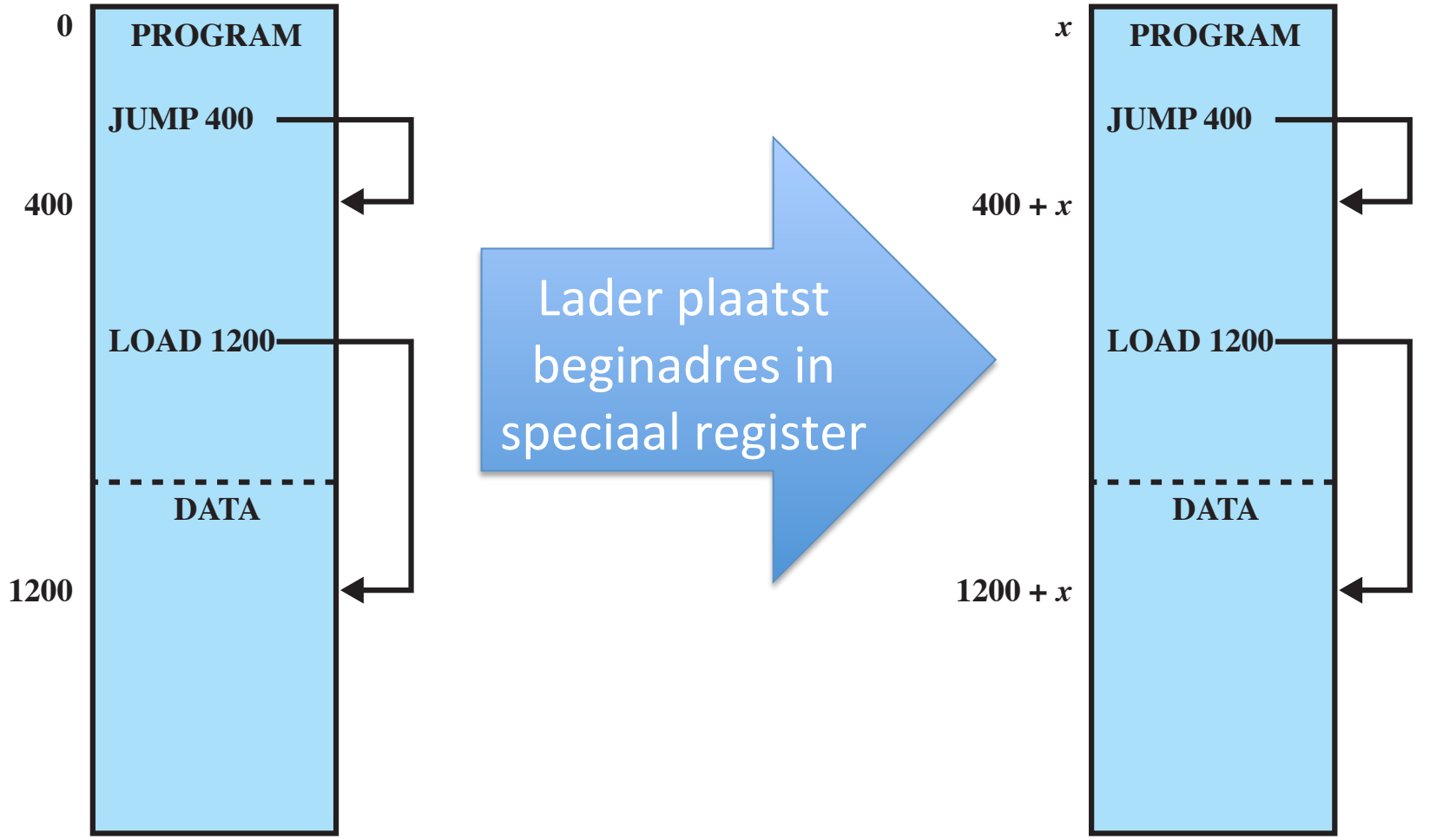
- Probleem: welke plaats in het RAM?
  - vaste plaats: eenvoudig, maar inflexibel
  - variabele plaats: vereist tweede relocatiestap

  - sommige besturingssystemen eisen dat programma alleen relatieve adressen bevat (.COM-bestand)

Voorbeeld: laden met relocatie



Voorbeeld:
laden met relatieve adressen



programmabestand

programma in RAM

Library

- samenvatting van meerdere objectbestanden in één groot bestand
- handzamer voor de programmeur

- linker behandelt library net als objectbestand

Standaardlibrary

- voorgedefiniëerde functies van een compiler:
  niet echt ingebouwd,
  maar in de standaardlibrary

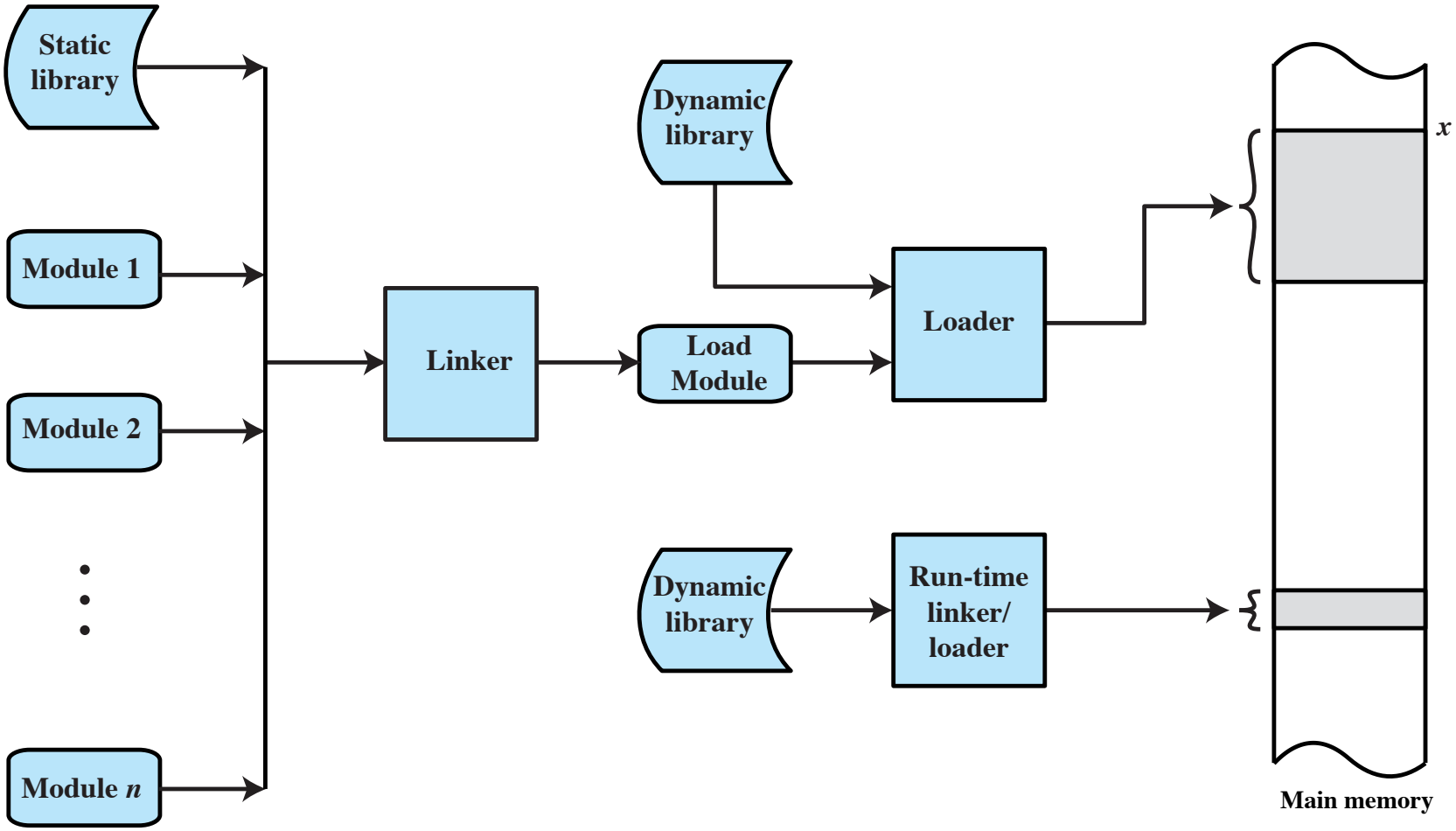
- standaardlibrary
  wordt bij veel programma's gebruikt
- elk programmabestand bevat dezelfde library
  – inefficiënt!

Dynamische library

- library in een apart bestand (.DLL) opgeslagen i.p.v. programmabestand
- laatste link-stap gebeurt pas
  - als programma geladen wordt
  - of als programma library-functie gebruikt

- vaak gebruikte libraries worden slechts één keer opgeslagen

Voorbeeld: van objectbestand tot proces



Samenvatting

- Assembly: gemakkelijk te onthouden notatie voor machine-instructies
- Assembler vertaalt ook labels
- Linker voegt objectbestanden+libraries samen in programmabestand
- Lader kopieert programmabestand naar RAM
- Dynamische libraries: linken tijdens/na laden

Toegift:
control structures in assembly

if – then – else

volgorde van operanden verschilt: de 1e operand is het doel.

Pascal	assembly (practicum)	assembly (8086)

if (i = 3) then
  j := 1
else
  k := 2;
	READ [i], R2
	SUB 3, R2, R0
	J.NZ else
	LOAD 1, R2
	WRITE R2, [j]
	J endif
	else: LOAD 2, R2
	WRITE R2, [k]
	endif:
		MOV AX, [i]
		CMP AX, 3
		JNZ else
		MOV [j], 1
		JMP endif
		else: MOV [k], 2
		endif:

loop

Pascal	assembly (practicum)	assembly (8086)

for i := 1 to 5 do
  ...;
	LOAD 1, R1
	WRITE R1, [i]
	for: READ [i], R1
	SUB 5, R1, R0
	J.L endfor
	...
	READ [i], R1
	ADD 1, R1, R1
	WRITE R1, [i]
	J for
	endfor:
		MOV [i], 1
		for: MOV AX,[i]
		CMP AX, 5
		JG endfor
		...
		MOV AX, 1
		ADD [i], AX
		JMP for
		endfor:

Inleveren

- alle onderdelen (.hds, .sym-bestanden) die nodig zijn
- verslag: bevat datastroomdiagram (zonodig verbeterd)
- assembly-programma (zonodig verbeterd)

Troostend slotwoord

Ik begrijp wel
dat niet iedereen een werkende processor krijgt,
maar ik ben ervan overtuigd
dat jullie zo meer geleerd hebben
dan wanneer ik slechts
een werkende processor had gedemonstreerd.